Proceedings of Machine Learning Research vol vvv:1–24, 2024

Data-Driven Modeling and Verification of Perception-Based Autonomous Systems

Thomas Waite WAITET@RPI.EDU
Dept. of Computer Science, Rensselaer Polytechnic Institute

Alexander Robey AROBEY1@SEAS.UPENN.EDU
Dept. of Electrical and Systems Engineering, University of Pennsylvania

Hamed Hassani HASSANI@SEAS.UPENN.EDU
Dept. of Electrical and Systems Engineering, University of Pennsylvania

George J. Pappas PAPPASG@SEAS.UPENN.EDU
Dept. of Electrical and Systems Engineering, University of Pennsylvania

Radoslav Ivanov IVANOR@RPI.EDU
Dept. of Computer Science, Rensselaer Polytechnic Institute

Abstract

This paper addresses the problem of data-driven modeling and verification of perception-based autonomous systems. We assume the perception model can be decomposed into a canonical model (obtained from first principles or a simulator) and a noise model that contains the measurement noise introduced by the real environment. We focus on two types of noise, benign and adversarial noise, and develop a data-driven model for each type using generative models and classifiers, respectively. We show that the trained models perform well according to a variety of evaluation metrics based on downstream tasks such as state estimation and control. Finally, we verify the safety of two systems with high-dimensional data-driven models, namely an image-based version of mountain car (a reinforcement learning benchmark) as well as the F1/10 car, which uses LiDAR measurements to navigate a racing track.

Keywords: safe autonomy; verification of perception models, neural network verification.

1. Introduction

From self-driving cars (Waymo) to taxi helicopters (VoloCity), the last few years have seen the development of a number of impressive autonomous systems. As the complexity of these systems increases, however, so does the concern for their safety. In fact, we have already witnessed accidents involving systems as diverse as autonomous cars (e.g., see the reports by the NHTSA (b) and the NTSB), chess-playing robots (as reported by The Guardian) and autonomous aircraft (analyzed in an FAA report). Furthermore, the United States government recently recorded 367 crashes involving autonomous cars over a 10-month period (NHTSA (a)). In order to prevent such incidents, it is essential that we verify the safety of autonomous systems before they are deployed in the wild.

Unlike classical control systems, modern autonomous systems introduce an extra layer of complexity since they operate in complex environments, which are perceived through high-dimensional measurements such as LiDAR scans and camera images. In turn, these measurements are processed by neural networks (NNs) used for estimation and control. Verifying such a closed-loop system at design-time poses two significant challenges: 1) environment models are difficult to develop from first principles due to unexpected noise, e.g., reflected LiDAR rays (Ivanov et al. (2020a)); 2) NNs

are not robust to even small input perturbations (Szegedy et al. (2013)) and distribution shifts (Recht et al. (2019)), which may reduce the utility of verification performed against an imperfect model.

To overcome these challenges, in this paper we propose a compositional verification approach that uses data-driven environment models. We compose two types of models: 1) a canonical environment model (e.g., during daytime with perfect visibility) that is obtained from first principles or through a simulator; 2) a data-driven noise model that is trained on real data to augment the canonical model with data artifacts observed during real system operation (e.g., blurred images or reflected LiDAR rays). This approach is motivated by recent work by Katz et al. (2022) on developing (and verifying) canonical models from real observations as well as work by Wu et al. (2023) on using generative models for certifying the robustness of NNs to real-world distribution shifts. The compositional model has several benefits over a single monolithic model: 1) the canonical model need not be data-driven, thereby reducing the data requirements for training the perception model; 2) different noise models can be composed with the canonical model to capture diverse scenarios; 3) individual noise models can be smaller, thereby alleviating verification scalability challenges.

Training noise models presents an interesting challenge since some types of noise are easier to capture using generative models than other types. For example, continuous noise (e.g., blur or contrast) is fairly benign and is easily learned by a generative model. However, discontinuous noise (e.g., reflected LiDAR rays or lens flare) is more adversarial and cannot be perfectly learned using a continuous model. We handle these two cases separately: 1) for benign noise, we use generative models, e.g., variational autoencoders as introduced by Kingma and Welling (2013); 2) for adversarial noise (defined as high-frequency large deviation from the canonical model), we train classifiers that indicate which part of the canonical measurement is affected and can be replaced with the effect of the adversarial noise. In fact, our experiments suggest that it is exactly the discontinuous noise that causes the largest deviations in control performance as compared to the canonical environment.

To evaluate the compositional data-driven modeling approach, we present two verification case studies: 1) Mountain Car (MC), which is a reinforcement learning (RL) benchmark available in Gymnasium; 2) the F1/10 Car, which is a 1/10-scale autonomous racing car platform. In MC, we train an image-based control pipeline and verify that the car reaches the goal for a range of benign noises, including blur and contrast. In the F1/10 case, we use existing LiDAR traces (Ivanov et al. (2020a)) to train an adversarial noise classifier. The noise model's quality is evaluated on the downstream control task, by comparing the control output on real vs. modeled LiDAR scans. Finally, we verify that the F1/10 car, coupled with a robust denoiser, can safely navigate an environment under adversarial LiDAR noise that was shown to be correlated with crashes by Ivanov et al. (2020a). In both cases, we performed the verification using our tool, Verisig (Ivanov et al. (2019, 2020b, 2021)).

In summary, the contributions of this paper are as follows: 1) a compositional data-driven method for developing perception models used in verification; 2) a noise-specific approach that uses generative models for benign noise and classifiers for adversarial noise; 3) two verification case studies illustrating the benefit of data-driven modeling and verification.

Related work. Verification of standard (non-neural) control systems is a mature field. A classical approach to reachability is by using Hamilton-Jacobi methods (Mitchell (2002); Chen et al. (2018)). Another class of techniques approximate reachable sets using Taylor models (Makino and Berz (2003); Chen et al. (2012), ellipsoids (Althoff (2015)), and polytopes (Chutinan and Krogh (2003)). Finally, there also exist methods that cast the problem as a satisfiability modulo theory program (Gao et al. (2013); Kong et al. (2015)). More recently, a number of works were developed for open-loop verification of NNs, such as input-output robustness, e.g., works by Dutta et al. (2018); Ehlers

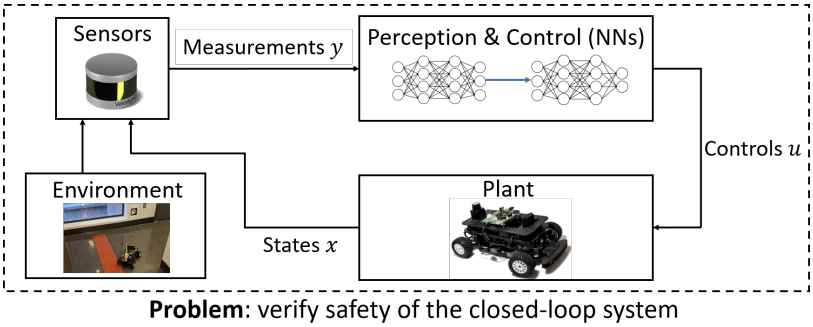


Figure 1: Overview of the problem considered in this paper.

(2017); Fazlyab et al. (2019); Gehr et al. (2018); Katz et al. (2017); Wang et al. (2018); Weng et al. (2018); Tran et al. (2020); Wang et al. (2021). Methods also exist for verification of closed-loop systems with NN components such as those by Ivanov et al. (2019); Huang et al. (2019); Dutta et al. (2019); Sun et al. (2019); Tran et al. (2019); Bogomolov et al. (2019); Dreossi et al. (2019).

For systems with high-dimensional perception, Katz et al. (2022) developed a verification method by training a generative image model using simulated data. Ivanov et al. (2020a) used a first-principles LiDAR model to verify the safety of the F1/10 car as it navigates a racing track. Hsieh et al. (2022) verified perception-based systems using approximate abstractions of the perception system. Although these works are good first steps, they do not account for the distribution shift between the canonical model and the real data that would be captured by our noise model. Approaches also exist (Dawson et al. (2022); Robey et al. (2020b, 2021)) for learning safe barrier certificates from images but these do not provide worst-case guarantees on the learned certificate. Finally, a number of works exist (Hanspal and Lomuscio (2023); Robey et al. (2020a); Wu et al. (2023)) that use model-based generative NNs for open-loop verification and robustness purposes; we borrow ideas from these approaches (e.g., model-based training) when tackling the closed-loop problem.

2. Problem Formulation

This section formalizes the problem addressed in this paper. We consider a closed-loop autonomous system, as illustrated in Figure 1. Formally, we are given a dynamical system of the sort:[1]

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k) = g_n(x_k, \delta_k) \circ g_c(x_k) \\ u_k &= h(y_k), \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the system state (e.g., position, velocity), $y \in \mathbb{R}^m$ are the measurements (e.g., camera images and LiDAR scans) and $u \in \mathbb{R}^p$ are the controls. The dynamics f are known. The observation model g is unknown but we assume it can be represented as the composition of a canonical environment model, g_c, and a noiser, g_n. The canonical model is known (either developed from first principles or a simulator); the noiser is unknown as it contains the noise profile of the real environment the system operates in; the parameter δ_k specifies the noise intensity (e.g., blur level). Finally, h encodes the perception/control pipeline and contains one or more NNs.

To train the noise model, we assume we are given a training set $\mathcal{D} = \{(x_i, y_i)\}$ of states and measurements. The training set is collected by running the system (e.g., manually) in the real environment. This setting is inspired by the one considered by Dean et al. (2020), where $\mathcal{D}$ is

1. A continuous-time formulation can also be handled by the proposed framework.

(*a*) Canonical (*b*) Reduced Contrast (*c*) Blur (*d*) Contrast + Blur

Figure 2: Examples of benign noise in the MC environment.

used to obtain probably approximately correct (PAC) bounds on perception error. An important challenge is developing a metric for evaluating the noise model's quality; metrics defined on the high-dimensional measurement space may be misleading, as distances (e.g., pixel differences) may not be semantically meaningful. Thus, part of the problem is to develop a semantically meaningful metric to evaluate the noise model. We now state the two problems considered in this work.

Problem 1 *Consider the closed-loop system in* (1)*. Given a training dataset,* $\mathcal{D} = \{(x_i, y_i)\}$*, train a noise model,* g_n*. Furthermore, develop a semantically meaningful evaluation metric for* g_n*.*

Problem 2 *Consider the closed-loop system in* (1) *where* g_n *is well trained. Given an initial set* $\mathcal{X}_0$*, the problem is to verify a safety property* ϕ *(e.g., no collisions) of the reachable states* x_k*:*

$$(x_0 \in \mathcal{X}_0) \Rightarrow \phi(x_k), \ \forall k \geq 0. \tag{2}$$

3. Motivating Examples

This section presents two examples to illustrate different system setups, i.e., an image-based vs. a LiDAR-based system, as well as different noise types, namely benign vs. adversarial. Finally, while the MC example is purely synthetic, the F1/10 case study uses real data to train the noise model.

3.1. Mountain Car: An Image-Based System Operating Under Benign Noise

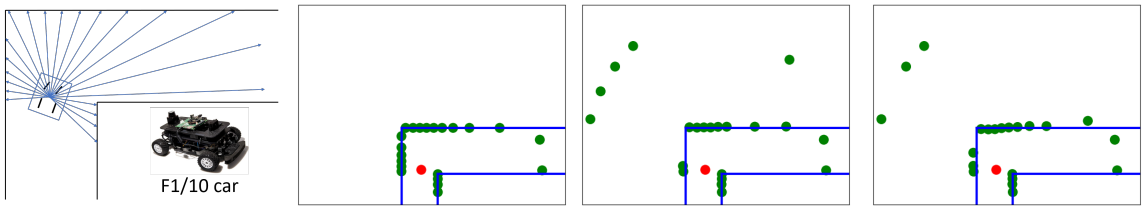
MC is an RL benchmark where the task is to drive an underpowered car up a hill, as illustrated in Figure 2(*a*) (we modified the environment by reducing the image size and making the car bigger). Since the car does not have enough power, it needs to learn to drive up the left hill so as to gather momentum and get to the goal on the right. MC consists of two states, position and velocity, which are both observed by the controller in the original MC environment. In this paper, we consider the case where the controller observes an image (such as the one in Figure 2(*a*)), coupled with velocity.[2]

Given an image-based controller, we are interested in verifying the system's robustness to environment noise. As illustrated in Figures 2(*b*) and 2(*c*), we consider two types of noise, contrast and blur (as well as their combination, shown in Figure 2(*d*)), that aim to capture realistic operating conditions. Different contrast corresponds to changing lighting conditions, e.g., time of day, whereas blur occurs at higher driving speeds. Both noise types are continuous and are classified as benign. Section 4 presents a method to train and verify a generative model for such types of noise.

3.2. The F1/10 Car: A LiDAR-Based System Operating Under Adversarial Noise

The F1/10 Car is a popular autonomous racing platform. It exhibits a number of the challenges introduced by full scale autonomous cars, such as noisy measurements, adversarial agents, and fast-paced environments. We focus on the case of a single car navigating a square track using LiDAR

2. As it is impossible to infer velocity from a single image, we leave the case of using multiple images for future work.



(*a*) LiDAR illustration (*b*) Canonical scan at x_k (*c*) Real scan at x_k (*d*) Real scan at x_{k+1}

Figure 3: Illustration of the F1/10 car, including canonical and real scans featuring adversarial noise.

measurements, as shown in Figure 3(*a*). In prior work, Ivanov et al. (2020a) used RL to train a number of controllers in a simulated environment and subsequently verified that these controllers are safe in the simulated environment. However, Ivanov et al. (2020a) also demonstrated that the controllers resulted in a number of accidents in the real world, as caused by noisy real data.

A canonical 21-dimensional LiDAR scan[3] for a right-hand turn is illustrated in Figure 3(*b*). Figures 3(*c*) and 3(*d*) show noisy real scans collected by Ivanov et al. (2020a) in a real environment similar to the simulated one; the car pose is the same (up to estimation error) in both environments. The real data contains reflected LiDAR rays (e.g., due to metallic surfaces), which manifest as perceived gaps in the environment. Such patterns are discontinuous and are classified as adversarial. Indeed, Ivanov et al. (2020a) showed that the number of accidents is significantly reduced if reflected surfaces are covered, thereby confirming the adversarial nature of LiDAR noise. Section 4 presents a classifier-based approach for modeling adversarial noise for verification purposes.

4. Modeling Framework

This section presents our modeling framework, including the canonical environment model and the two types of noise models. We also present a method for evaluating the quality of these components.

4.1. Canonical Model

We focus on static environments where a first-principles model can describe the main features but may fail to capture environmental noise such as reflective surfaces and changing lighting conditions. Building a (canonical) first-principles model is particularly useful in such a setting as it enables the development of controllers in simulation. Once such a controller is built, deploying the system in the real environment requires only a "de-noiser" for that specific environment, i.e., an additional component that is trained to translate real measurements to canonical ones. Thus, canonical models are by definition modular – the same canonical model can be be composed with different noiser/de-noiser settings and can thus be used for different environments that are the same up to measurement noise (e.g., roads with various degrees of marking degradation). In future work, we will also consider the case where the canonical model is augmented with other agents observed in the real environment.

4.2. Noise Model

When developing a noise model for the purposes of verification, there are two main scalability-related requirements: 1) models should be small and modular; 2) models should not introduce significant spurious (unrealistic) noise, as that greatly complicates the verification task. Training a single model that satisfies both of these requirements is challenging, especially when one considers the types of noise illustrated in Section 3. In particular, such a model would be exceedingly large

3. A typical scan consists of 1081 rays, but we use a subsampled version to alleviate verification scalability challenges.

for verification purposes and would likely introduce a large number of spurious behaviors in order to capture discontinuous noise (NNs are by definition continuous, so the only way to capture discontinuous dynamics is through including all in-between values as well). Thus, we argue that it is necessary to train a separate noise model for each type of noise. In what follows, we define each type of noise formally and describe a training procedure for the corresponding noise model.

Benign Noise. Intuitively, benign noise is noise that is easy to learn, both to generate and to be robust to. In their seminal paper, Szegedy et al. (2013) show that NN classifiers are robust to Gaussian noise, even if the noise results in more distorted images than adversarial noise. We observe a similar pattern in the context of the examples considered in this paper: continuous/benign noise is easier to learn than discontinuous/adversarial noise. Formally, we define benign noise as follows.

Definition 1 (Benign Noise) *Let $\Delta_x = g_n(x, g_c(x), \delta) - g_c(x)$ be the measurement noise introduced by the environment. We classify Δ_x as benign if it changes slowly across the state space $\mathcal{X}$:*

$$\forall \varepsilon_x > 0 \; \exists \varepsilon_\Delta > 0 \; \textit{s.t.} \; \forall x, x' \in \mathcal{X}, \; \|x - x'\|_\infty \leq \varepsilon_x \Rightarrow \|\Delta_x - \Delta_{x'}\|_\infty \leq \varepsilon_\Delta.$$

Definition 1 captures not only noise that is small in magnitude, but also noise that may be large in magnitude but is not changing across the state-space (e.g., a change in lighting conditions may result in large noise overall but is easy to learn as it is reasonably constant). This definition captures both types of noise considered in the MC case study, which introduce significant changes to the canonical image but are consistent across states. To train a generative model for benign noise given a training set $\mathcal{D} = \{(x_i, y_i)\}$, one can use a number of losses, e.g., mean squared error, binary cross entropy (BCE), or reconstruction loss as introduced by Kingma and Welling (2013). In the experiments, we use BCE with fully-connected NN models as this is the NN architecture supported by Verisig.

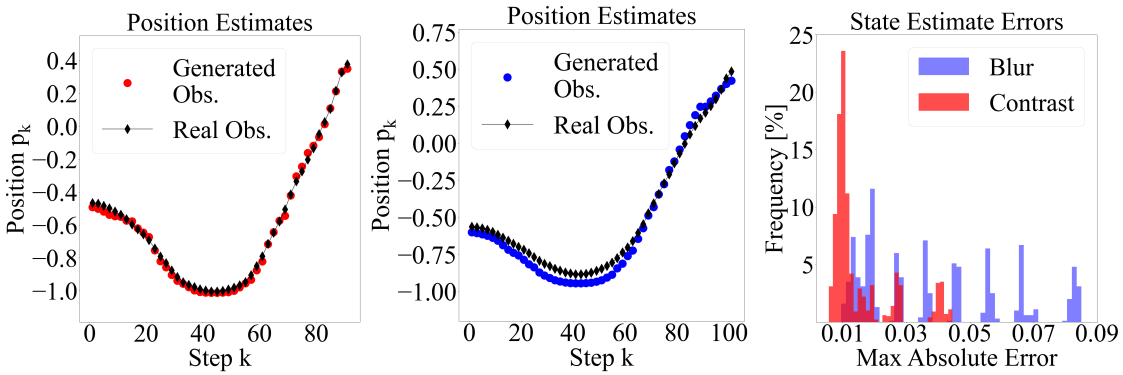
Adversarial Noise. Adversarial noise is high-frequency noise that may change quickly across the state space, as demonstrated in Figures 3(*c*) and 3(*d*). For example, LiDAR rays are always reflected be some surfaces (or may be reflected by other surfaces only under certain angles). Such noise is similar to the adversarial noise for classification tasks discovered by Szegedy et al. (2013).

Definition 2 (Adversarial Noise.) *We classify Δ_x (introduced in Definition 1) as adversarial noise if there exists a subset $\mathcal{X}_a \subseteq \mathcal{X}$ such that $\forall x \in \mathcal{X}, \forall x_a \in \mathcal{X}_a, \; \|\Delta_x - \Delta_{x_a}\|_\infty > \varepsilon_a$ for some $\varepsilon_a > 0$.*

As argued above, training a generative model for adversarial noise is challenging since we are effectively trying to capture a discontinuous space using a continuous model. Thus, we propose to use a classifier, $c_n : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}^m$, that determines whether a measurement (dimension) is corrupted by adversarial noise. If a certain dimension is flagged, then its value is reset to a default value, y_a, corresponding to the effect of that type of noise (e.g., reflected LiDAR rays are set to the maximum LiDAR range whereas pixels affected by glare can be reset to white). Formally,

$$g_n(x, g_c(x))^d = \begin{cases} g_c(x)^d & \text{if } c_n(x, g_c(x))^d = 0 \\ y_a & \text{if } c_n(x, g_c(x))^d = 1, \end{cases} \tag{3}$$

where $g_c(x)^d$ is the value in position d of $g_c(x)$ (same for $g_n(x, g_c(x))^d$). Thus, c_n learns which states are likely to result in adversarial noise. To train c_n, we transform the training set $\mathcal{D} = \{(x_i, y_i)\}$ into a classification set $\mathcal{D}_c = \{((x_i, y_i), n_i)\}$ where n_i is a binary vector of labels indicating which dimensions of y_i contains adversarial noise. To generate labels, we set $n_i^d = 0$ if $|y_i^d - g_c(x)^d| \leq t_\Delta$ and $n_i^d = 1$, otherwise (for a user-defined threshold t_Δ). If $\mathcal{D}$ is dense enough to identify the adversarial subset $\mathcal{X}_a$, then one could use a clustering algorithm to obtain more refined labels n_i. Since such a procedure was not necessary for our case studies, we leave this analysis for future work.



(*a*) Low contrast ($\delta_c = 0.3$) (*b*) High blur ($\delta_b = 0.3$) (*c*) Histogram of estimate errors

Figure 4: Evaluation of generative noise models on MC. Figures (a) and (b) show estimated position trajectories (using modeled vs. real observations) in two challenging noise scenarios. Figure (c) shows absolute position estimation errors over 1000 simulated trajectories.

4.3. Model Evaluation

Since both noise modeling approaches discussed in this section are data-driven, a statistical evaluation is required as well. We evaluate the performance of the trained model on a labeled test set $\mathcal{D}_t = \{(x_i, y_i)\}$. Although a straightforward evaluation would be to compute the average error norm $\|y_i - g_n(x_i, g_c(x_i), \delta_i)\|$, such an evaluation may be misleading since element-wise distances in high-dimensional measurement spaces (e.g., images) are not semantically meaningful. For example, the model may capture the adversarial noise in an adjacent pixel/ray – while such a model is not perfect, it may elicit the same overall system behavior. Thus, we evaluate the noise model on the downstream task (e.g., control or estimation), where distances are semantically meaningful.

We propose three evaluation metrics depending on the downstream task. The *state-estimation-based metric* applies to systems with a state estimator (e.g., MC); in this case, we calculate the average error $\|h_e(y_i) - h_e(g_n(x_i, g_c(x_i), \delta_i))\|$ over the test set, where h_e is a state estimator. Similarly, a *control-based metric* calculates the average error of a controller h_c: $\|h_c(y_i) - h_c(g_n(x_i, g_c(x_i), \delta_i))\|$. Finally, a *trajectory-based metric* compares the number of unsafe events recorded in the test set vs. the number of unsafe events recorded by simulating the system using the noiser model. The first two metrics can also be used to generate a PAC bound on the state estimation/control error. In future work, we will incorporate this bound in the verification problem as well.

5. Verification

Given models for all system components, we can perform verification using any verification tool for autonomous systems with NN components. We use our tool Verisig as it has shown great scalability on the examples considered in this paper. Verisig focuses on fully-connected NNs with smooth activations (e.g., sigmoid and tanh) and works by transforming the NN into a hybrid system that is then composed with the plant's dynamical system. The resulting hybrid system reachability problem is solved by the tool Flow* (Chen et al. (2012)), which uses Taylor Model approximations.

6. Case Studies

This section presents two case studies, based on the motivating examples from Section 3. A more detailed description is available in the appendices.

System	Initial Conditions	Total Intervals	Avg. Time [s]	Avg. NN Time [s]	Avg. Branches
MC: Contrast	$p_0 \in [-.59, -.4]$ $\delta_c \in [0.3, 2]$	816	48396	26259	1.26
MC: Blur	$p_0 \in [-.59, -.4]$ $\delta_b \in [0, 0.3]$	2057	62476	29702	1.02
MC: Composite	$p_0 \in [-.55, -.4]$ $\delta_c \in [0.5, 2]$ $\delta_b \in [0, 0.15]$	1272	61395	32894	1.38
F1/10	$x_{1,0} \in [-0.1, 0.1]$ $x_{2,0} \in [1, 1]$ $\theta_0 \in [0, 0]$	8236	10733	4482	6.88

Table 1: Verification results for both case studies. The last column shows the number of branches in the composed system, e.g., due to the adversarial noise classifier possibly outputting two values due to uncertainty. Branches present a scalability challenge as each one needs to be verified separately.

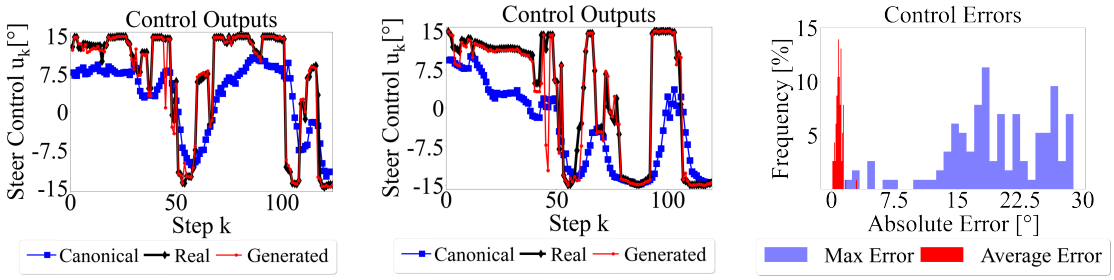
6.1. Mountain Car

As described in Section 3, in MC the controller observes an image and ground-truth velocity. The car's dynamics are: $p_{k+1} = p_k + v_k$, $v_{k+1} = v_k + 0.0015u_k - 0.0025 * cos(3p_k)$, where p and v are position and velocity, respectively, and $u \in [-1, 1]$ is thrust. The initial condition $p_0 \in [-0.6, -0.4]$ is at the bottom of the mountain. We would like to verify the system's robustness to two types of noise, contrast and blur. Since images are produced by a black-box simulator, we train a generative canonical observation model, described next. The control pipeline consists of two parts: 1) a state estimator, h_s, that estimates the car position from an image (explained next); 2) a controller, h_c, that takes the position estimate and ground-truth velocity as inputs. The controller is a pre-trained and pre-verified (for $p_0 \in [-0.59, 0.4]$) NN, borrowed from Ivanov et al. (2019).

Model Training. To train all models, we create a training set $\mathcal{D} = \{(p_i, y_i^c, y_i^{ct}, y_i^b)\}$ of 82000 examples. We evenly sample 2000 positions and canonical images (grayscaled and downsampled to 20×30), y_i^c; for each one, we create 40 images with varying contrast, y_i^{ct}, and blur, y_i^b. The state estimator, $h_e : \{y_i^c, y_i^{ct}, y_i^b\} \mapsto p_i$, is trained using least squares on all images. The canonical model, $g_c : p_i \mapsto y_i^c$, is trained using BCE. The contrast model, $g_{ct} : y_i^c \times \delta_{c,i} \mapsto y_i^{ct}$, and the blur model, $g_b : y_i^c \times \delta_{b,i} \mapsto y_i^b$, map a canonical image to a noised image, for a range of contrast, $\delta_{c,i} \in [0.3, 2.0]$, and blur, $\delta_{b,i} \in [0, 0.3]$. All models are fully-connected NNs with tanh activations with the following shapes (notation $[a_1, \ldots, a_n]$ means the NN has n layers with a_i neurons each; if a_i are the same, we write $n \times a$): 1) $h_e : 3 \times 20$; 2) $g_c : [50, 100]$; 3) $g_{ct} : 1 \times 50$; 4) $g_b : 1 \times 50$.

Model Quality. To evaluate models, we use the state-estimation-based metric discussed in Section 4.3. We compare state estimates on 2000 test trajectories (with varying noise levels) using: 1) ground truth images vs. 2) images produced by the data-driven model. Figures 4(*a*) and 4(*b*) show estimated trajectories under the most challenging scenarios of low contrast and high blur conditions, respectively. As can be seen in the figures, both noise models result in very low difference in estimation performance. Figure 4(*c*) summarizes the absolute errors over all trajectories. All errors are bounded within a small range; although blur errors have a longer tail, the error is bounded by 0.09, which corresponds to 5% of the total position space. It is important to note that one can even compose the blur and contrast models to achieve more complex noise patterns. Although not shown in the interest of space, the compositional model achieves similar performance on restricted domain for noise parameters (namely, $\delta_c \in [.5, 2]$ and $\delta_b \in [0, 0.15]$). Based on these promising results, we conclude that the noise models are sufficiently accurate so that a verification result is meaningful.

Verification. We verify that the car goes up the hill (with a reward of at least 90, which was the property verified by Ivanov et al. (2019) for the original controller) for a range of initial positions and noises, as shown in Table 1. All ranges were verified safe. The verification was carried out in parallel using sub intervals (of size 0.00025) of the initial state space and the noise range. If verification of any given interval did not terminate within 24 hours (we used a server with 95 cores and 500GB of RAM) the process was restarted with five sub-intervals (and restarted again if necessary).



(*a*) Moderate max error (*b*) High max error at step 47 (*c*) Histogram of control errors

Figure 5: Control-based evaluation of F1/10 noise model, using controller c_2. Figures (*a*) and (*b*) compare control outputs on two trajectories for canonical, noised, and real measurements. Figure (*c*) summarizes the maximum and mean errors over all 115 real trajectories.

6.2. The F1/10 Car

The F1/10 case study is significantly more challenging than MC since it is based on real data, and it includes substantial adversarial noise. We use real data traces collected by Ivanov et al. (2020a) that were designed to evaluate the simulation-to-reality (sim2real) gap between the modeled and the real system. We aim to bridge the sim2real gap by training a data-driven noise model, as well as a "denoiser" that filters adversarial noise in order to use the original controller. We first train high-quality noise and denoiser models, before verifying the car safely navigates the track.

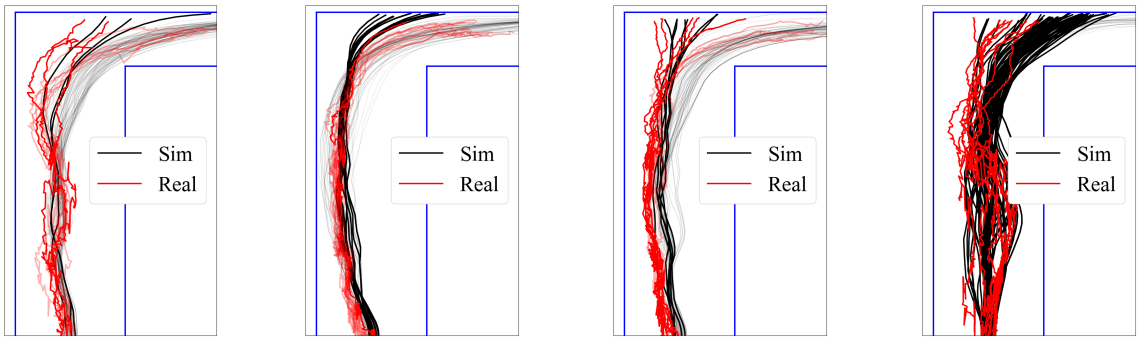
Noiser Training Results					
Model Size	3x100	4x100	5x100	6x100	6x200
WSA [%]	61	77	87	98	100
Control Err.	3700	2312	1485	774	594
Safe Trajectory Distribution [%]					
Real Data	**Simulated Data Using Noiser Model**				
Controller c_1: **80**	88	83	88	82	97
Controller c_2: **90**	75	80	92	77	74
Controller c_3: **90**	85	78	98	91	91
Controller c_4: **0**	29	29	29	32	27

Table 2: Training results and crash distributions for different noise models. Pre-trained controllers and real data were borrowed from Ivanov et al. (2020a) (controllers c_1, c_2, c_3, and c_4 correspond to controllers "DDPG,64,C3", "TD3,64,C2", "TD3,128,C2", and "DDPG,128,C3").

Model Training. Since the data traces do not contain ground truth car poses, we develop a state estimator using a particle filter (Thrun et al. (2005)), based on the bicycle and LiDAR models in Ivanov et al. (2020a). Using the state estimates, we create a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}_i = (x_{1,i}, x_{2,i}, \theta_i)$ denotes the car's (x, y)-coordinates and orientation, respectively, and y_i is the corresponding LiDAR scan. To train the LiDAR noise model, we label each LiDAR scan according to the procedure in Section 4. Given a canonical scan $g_c(\mathbf{x}_i)$, we set $n_i^d = 1$ if $|g_c(\mathbf{x}_i)^d - y_i^d| > t_\Delta^*$, and $n_i^d = 0$, otherwise.[4] Using the labeled training set $D_l = \{((g_c(\mathbf{x}_i), \mathbf{x}_i), n_i)\}$, we train classifiers of increasing sizes (listed in Table 2) using weighted BCE (to account for class imbalance between adversarial and non-adversarial examples). Training multiple noise models allows us to: 1) demonstrate that the task is feasible; 2) identify the largest model that can be handled by Verisig. Finally, we train a 1×100 fully-connected denoiser NN, $h_d : y_i \mapsto \hat{y}_i$, on pairs $(g_c(\mathbf{x}_i), y_i)$.

Model Quality. Since training data is limited (115 trajectories), we train on all data and do not have a test set (in future work, we will perform an exhaustive cross-validation study). To get around this issue, we use both the control-based metric and the trajectory-based metrics discussed in Section 4.3. Table 2 shows an evaluation of all trained noise models according to whole scan accuracy

4. The threshold is chosen to minimize the error between noised canonical scans and real scans over the training data.



(*a*) Controller c_1 (*b*) Controller c_2 (*c*) Controller c_3 (*d*) Controller c_4

Figure 6: Simulated trajectories under the 5x100 noiser model vs. real trajectories for four different controllers (without using a denoiser). Bold trajectories indicate those that ended in a crash.

(WHA); WHA is preferred to individual accuracy as even a single adversarial ray can significantly affect control output. The table shows that WHA is directly proportional to lower error between control outputs on the model vs. real scans. To evaluate the model according to the control-based metric, we use four pre-trained and pre-verified controllers from Ivanov et al. (2020a), labeled $c_1, \ldots, c_4$. Figures 5(*a*) and 5(*b*) show the control outputs, when using c_2, on two example real trajectories. We note that the adversarial noise model is very accurate, especially compared to the canonical model, which produces significantly different control outputs. Furthermore, Figure 5(*c*) shows the maximum and average control errors over all trajectories. Unlike MC, the error is not always bounded and is sometimes equal to the full control range of 30 degrees. At the same time, the average error is quite small, indicating great model performance on average.

Finally, we evaluate the model in terms of the trajectory-based metric, which effectively serves as a test set since the simulated trajectories deviate somewhat from the car states observed in real data. The bottom rows of Table 2 compare the number of crashes observed in the real data vs. the number of crashes observed in simulation for the same controllers (without using the denoiser). Not only are the number of crashes very close, but we also demonstrate qualitatively in Figure 12 that the crashing trajectories observed in the real experiments look very similar to the simulated ones when the data-driven noise model is used. These results suggest that the learned LiDAR noise model is a good approximation of real LiDAR noise and is thus an appropriate model for verification purposes.

Verification. Having established the accuracy of the data-driven noise model, we now verify that the car can safely navigate the noisy environment when also equipped with a denoiser. Using controller c_2 and the 5x100 noiser model, we verify a that if the car starts in the range $x_1 \in [-0.1, 0.1]$ in the middle of the hallway, then it can safely navigate the right-hand turn (a similar property was verified by Ivanov et al. (2020a)). The verification statistics are reported in Table 1. Similar to MC, we split the initial range into sub-intervals and iteratively verify the safety of each initial interval.

7. Conclusion

This paper addressed the problem of data-driven modeling and verification of perception-based autonomous systems. We proposed a compositional modeling approach by using a first-principles canonical model, coupled with a data-driven model capturing the real environment's noise, both in the case of benign and adversarial noise. We presented two cases studies illustrating the challenges of training and verifying such a compositional model. Future work includes performing an experimental evaluation that demonstrates the benefit of verification on a real platform. Futhermore, we will investigate the development of noise models that include additional agents in the environment.

References

Matthias Althoff. An introduction to cora 2015. *ARCH@ CPSWeek*, 34:120–151, 2015.

Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. Juliareach: a toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 39–44, 2019.

Mo Chen, Sylvia L Herbert, Mahesh S Vashishtha, Somil Bansal, and Claire J Tomlin. Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3675–3688, 2018.

Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *2012 IEEE 33rd Real-Time Systems Symposium*, pages 183–192. IEEE, 2012.

Alongkrit Chutinan and Bruce H Krogh. Computational techniques for hybrid system verification. *IEEE transactions on automatic control*, 48(1):64–75, 2003.

Alex Clark. Pillow (pil fork) documentation, 2015. https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods. *arXiv preprint arXiv:2202.11762*, 2022.

Sarah Dean, Nikolai Matni, Benjamin Recht, and Vickie Ye. Robust guarantees for perception-based control. In *Learning for Dynamics and Control*, pages 350–360. PMLR, 2020.

Tommaso Dreossi, Alexandre Donzé, and Sanjit A Seshia. Compositional falsification of cyber-physical systems with machine learning components. *Journal of Automated Reasoning*, 63:1031–1053, 2019.

S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, pages 121–138. Springer, 2018.

S. Dutta, X. Chen, and S. Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 157–168. ACM, 2019.

R. Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

F1/10 Car. F1/10 Autonomous Racing Competition. http://f1tenth.org.

FAA. Federal Aviation Administration: Summary of the FAA's Review of the Boeing 737 MAX. https://www.faa.gov/foia/electronic_reading_room/boeing_reading_room/media/737_RTS_Summary.pdf.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *arXiv preprint arXiv:1906.04893*, 2019.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Sicun Gao, Soonho Kong, and Edmund M Clarke. dreal: An smt solver for nonlinear theories over the reals. In *Automated Deduction–CADE-24: 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings 24*, pages 208–214. Springer, 2013.

T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. AI2: Safety and robustness certification of neural networks with abstract interpretation. In *Security and Privacy (SP), 2018 IEEE Symposium on*, 2018.

Gymnasium. Mountain Car. https://gymnasium.farama.org/environments/classic_control/mountain_car_continuous/.

Harleen Hanspal and Alessio Lomuscio. Efficient verification of neural networks against lvm-based specifications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3894–3903, 2023.

Chiao Hsieh, Yangge Li, Dawei Sun, Keyur Joshi, Sasa Misailovic, and Sayan Mitra. Verifying controllers with vision-based perception using safe approximate abstractions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4205–4216, 2022. doi: 10.1109/TCAD.2022.3197508.

Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–22, 2019.

R. Ivanov, T. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Case study: Verifying the safety of an autonomous racing car with a neural network controller. In *International Conference on Hybrid Systems: Computation and Control*, 2020a.

Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178, 2019.

Radoslav Ivanov, Taylor J Carpenter, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. Verifying the safety of autonomous systems with neural network controllers. *ACM Transactions on Embedded Computing Systems (TECS)*, 20(1):1–26, 2020b.

Radoslav Ivanov, Taylor Carpenter, James Weimer, Rajeev Alur, George Pappas, and Insup Lee. Verisig 2.0: Verification of neural network controllers using taylor model preconditioning. In *Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part I*, pages 249–262. Springer, 2021.

G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

Sydney M Katz, Anthony L Corso, Christopher A Strong, and Mykel J Kochenderfer. Verification of image-based neural network controllers using generative models. *Journal of Aerospace Information Systems*, 19(9):574–584, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dreach: δ-reachability analysis for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21*, pages 200–205. Springer, 2015.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Kyoko Makino and Martin Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 6:239–316, 2003.

Ian Michael Mitchell. *Application of level set methods to control and reachability problems in continuous and hybrid systems*. stanford university, 2002.

NHTSA. National Highway Traffic Safety Administration: Summary Report: Standing General Order on Crash Reporting for Level 2 Advanced Driver Assistance Systems, a. https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-06/ADAS-L2-SGO-Report-June-2022.pdf.

NHTSA. Us national highway traffic safety administration: Investigation pe 16-007, b. https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.pdf.

NTSB. US National Transportation Safety Board: Preliminary Report Highway HWY18MH010. https://www.ntsb.gov/investi-gations/AccidentReports/Reports/HWY18MH010-prelim.pdf.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

Alexander Robey, Hamed Hassani, and George J Pappas. Model-based robust deep learning: Generalizing to natural, out-of-distribution data. *arXiv preprint arXiv:2005.10247*, 2020a.

Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020b.

Alexander Robey, Lars Lindemann, Stephen Tu, and Nikolai Matni. Learning robust hybrid control barrier functions for uncertain systems. *IFAC-PapersOnLine*, 54(5):1–6, 2021.

X. Sun, H. Khedr, and Y. Shoukry. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 147–156. ACM, 2019.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, et al. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

The Guardian. Chess robot grabs and breaks finger of seven-year-old opponent. https://www.theguardian.com/sport/2022/jul/24/chess-robot-grabs-and-breaks-finger-of-seven-year-old-opponent-moscow.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. 2005.

H. Tran, F. Cai, D. M. Lopez, P. Musau, T. T. Johnson, and X. Koutsoukos. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Transactions on Embedded Computing Systems*, 18(5s):105, 2019.

H. Tran, S. Bak, W. Xiang, and T. T. Johnson. Verification of deep convolutional neural networks using imagestars. In *32nd International Conference on Computer-Aided Verification (CAV)*. Springer, July 2020.

VoloCity. VoloCity:The air taxi that's a cut above. https://www.volocopter.com/en/solutions/volocity.

S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, pages 6367–6377, 2018.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

Waymo. Waymo: The World's Most Experienced Driver. https://waymo.com/.

T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. Boning, and I. Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5273–5282, 2018.

Haoze Wu, Teruhiro Tagomori, Alexander Robey, Fengjun Yang, Nikolai Matni, George Pappas, Hamed Hassani, Corina Pasareanu, and Clark Barrett. Toward certified robustness against real-world distribution shifts. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 537–553. IEEE, 2023.

Appendix A. Mountain Car

This section provides a detailed explanation of the mountain car (MC) model and the verification task.

A.1. Dynamics

The MC dynamics are as follows:

$$p_{k+1} = p_k + v_k$$
$$v_{k+1} = v_k + 0.0015u_k - 0.0025 * cos(3p_k),$$

where p_k and v_k are position and velocity, respectively, with $p_0 \in [-0.6, -0.4]$. Note that v_k is constrained to be within $[-0.07, 0.07]$ and p_k is constrained to be within $[-1.2, 0.6]$; this means that the MC model is actually a hybrid system when those constraints are reached. The inputs $u_k \in [-1, 1]$ are thrust.

A.2. Controller

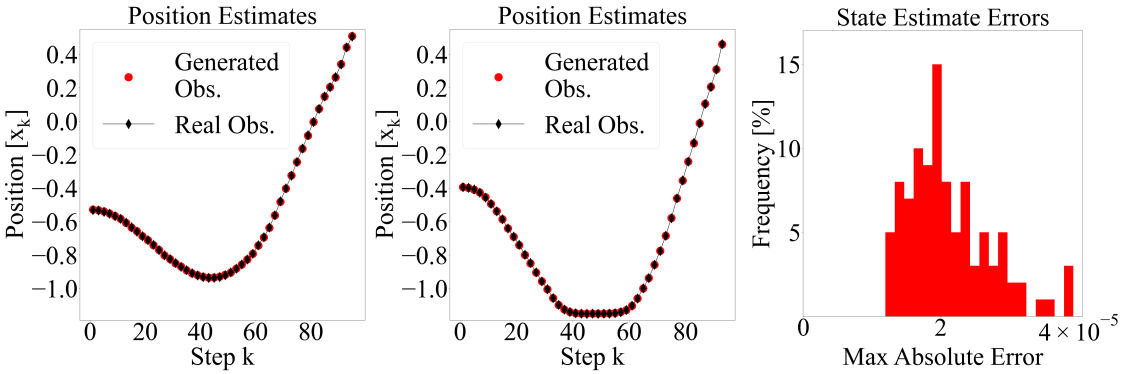
The controller, h_c, is a 2×16 NN with sigmoid activations in the hidden layers and tanh activation on the output neuron. This controller was verified to be safe by Ivanov et al. (2019), i.e., it reaches the goal with a reward of at least 90. Note that this controller takes position and velocity as input.

A.3. Training Set

In this paper we focus on the case where the control pipeline does not observe position and velocity but rather observes an image and ground-truth velocity. Furthermore, we consider two types of noise, contrast and blur, as illustrated in Figure 2. In order to train all models described below, we build a training set $\mathcal{D} = \{(p_i, y_i^c, y_i^{ct}, y_i^b)\}$ of 82000 examples. We evenly sample 2000 positions and corresponding canonical images (grayscaled, normalized to [0,1], and downsampled from the original $400 \times 600 \times 3$ to 20×30), y_i^c, as well as 20 contrasted, y_i^{ct}, and 20 blurred, y_i^b, images per position. Specifically, for each position p_i, the contrasted and blurred images are sampled evenly with $\delta_c \in [0.3, 2.0]$ and $\delta_b \in [0, 0.3]$. Contrast is added using the Python Image Library (PIL) ImageEnhance module where $\delta_c = 0$ produces a solid gray image, $\delta_c = 1$ produces the original image, and $\delta_c > 1$ produces a higher contrast version of the original image Clark (2015). Blur is added as follows: $y_i^b = 0.5g_c(p_i - \delta_{b,i}) + y_i^c + 0.5g_c(p_i + \delta_{b,i})$, i.e., a canonical image with a lighter overlay of left and right shifted images. Blurred images are then normalized to [0, 1].

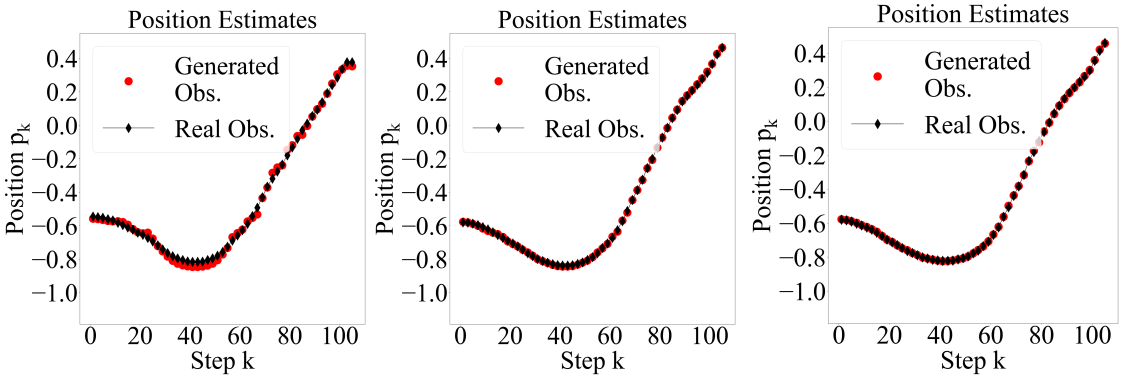
A.4. State estimator

The state estimator $h_e : \{y_i^c, y_i^{ct}, y_i^b\} \mapsto p_i$ takes an image as input and outputs position. The state estimator is trained using least squares on all images in $\mathcal{D}$ and on a set of 100000 synthetic composite images $\{g_b(g_c(y_i^c, \delta_{c,i}), \delta_{b,i})\}$. The state estimator is 3×20 NN with tanh activations on the hidden layers and a tanh activation on the output neuron. The output neuron is normalized to output only valid positions, values in the range $[-1.2, 0.6]$.



(*a*) Trajectory ($p_0 = -0.53$) (*b*) Trajectory ($p_0 = -0.4$) (*c*) Histogram of estimate errors

Figure 7: Evaluation of generative canonical model on MC. Figures (a) and (b) show estimated position trajectories (using modeled vs. real observations) in simulations. Figure (c) shows absolute position estimation errors over 100 simulated trajectories.



(*a*) Low contrast ($\delta_c = 0.3$) (*b*) Near canonical contrast ($\delta_c = 1.06$) (*c*) High contrast ($\delta_c = 2.0$)

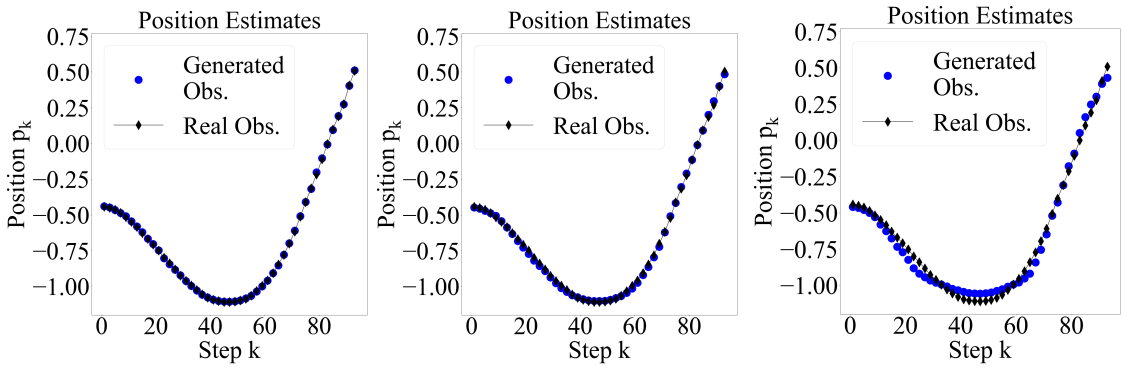
Figure 8: Additional evaluation of generative contrast models on MC. Figures show estimated position trajectories (using modeled vs. real observations) in simulations. Figure (a) shows a trajectory under low contrast, (b) shows a trajectory under little contrast change from the canonical environment, and (c) shows a trajectory under high contrast.

A.5. Canonical Model

Since the MC simulator is a black-box simulator, we do not have a first principles canonical model in this case. Thus, we use the canonical images in $\mathcal{D}$ to train a $[50, 100]$ generative model, $g_c : p_i \mapsto y_i^c$, that approximates the simulator. Figure 7 demonstrates the performance of the generative canonical model according to the state-estimation-based metric. As can be seen in the figures, the generative canonical model is a very good approximation of the simulator. The generative canonical model is trained using BCE.

A.6. Contrast Noise Model

The contrast model, $g_{ct} : y_i^c \times \delta_{c,i} \mapsto y_i^{ct}$, maps a canonical image to a noised image for a range of contrast intensities, $\delta_{c,i} \in [0.3, 2.0]$. The contrast model is 1×50 and is trained using BCE and weight decay.



(a) No blur ($\delta_b = 0$) (b) Moderate blur ($\delta_b = 0.15$) (c) High blur ($\delta_b = 0.3$)

Figure 9: Additional evaluation of generative blur models on MC. Figures show estimated position trajectories (using modeled vs. real observations) in simulations. Figure (a) shows a trajectory under no blur, (b) shows a trajectory under moderate blur, and (c) shows a trajectory under high blur.

A.7. Blur Noise Model

The blur model, $g_b : y_i^c \times \delta_{b,i} \mapsto y_i^b$, map a canonical image to a noised image, for a range of blur intensities, $\delta_{b,i} \in [0, 0.3]$. The blur model is 1×50 and is trained using BCE and weight decay.

A.8. Composite Model

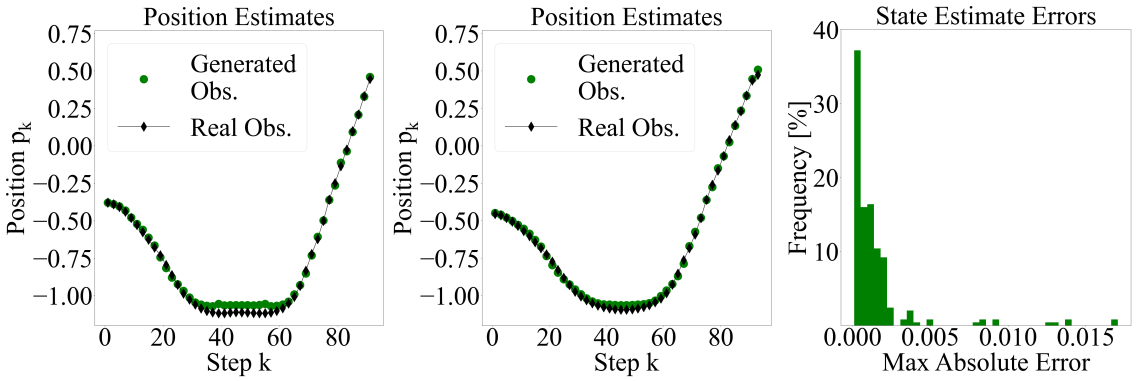
Interestingly, we do not train a separate composite model but rather compose the contrast and the blur models, $g_{ct}(\delta_{ct}) \circ g_b(\delta_b)$, in order to demonstrate that one can generate complex noise patterns by composing individual noise models. Figures 10 provide an evaluation of the composite model in terms of the state-estimation-based metric, on restricted domain for noise parameters (namely, $\delta_c \in [.5, 2]$ and $\delta_b \in [0, 0.15]$). Since the two models were only trained on canonical images, the composition quality deteriorates for larger noise intensities. The figures show the composite model performs similarly well to the single noise models such that composing individual models is a promising approach for modeling complex noise patterns.

A.9. Full System

The full closed-loop system is described below:

$$\begin{aligned} p_{k+1} &= p_k + v_k \\ v_{k+1} &= v_k + 0.0015u_k - 0.0025 * cos(3p_k) \\ y_k &= g(p_k) \\ \hat{p}_k &= h_e(y_k) \\ u_k &= h_c(\hat{p}_k, v_k), \end{aligned} \tag{4}$$

where $g = g_{ct}(\delta_{ct}) \circ g_c$ in the case of contrast noise, $g = g_b(\delta_b) \circ g_c$ in the case of blur noise, $g = g_b(\delta_b) \circ g_{ct}(\delta_{ct}) \circ g_c$ in the case of composite noise.



(*a*) Trajectory ($p_0 = -0.4$) (*b*) Trajectory ($x_0 = -0.59$) (*c*) Histogram of estimate errors

Figure 10: Evaluation of composite noise model on MC. Figures (a) and (b) show estimated position trajectories (using modeled vs. real observations) in simulations with noise parameters $\delta_c = 0.5$ and $\delta_b = 0.15$. Figure (c) shows absolute position estimation errors over 250 simulated trajectories

A.10. Verification

We would like to verify that the car reaches the goal with a reward of at least 90. The reward is $r_k = -0.1u_k^2$ for each step before the goal. If the car reaches the goal, $r_k = 100$. Since verifying liveness properties is undecidable in general, we formulate the property as a bounded liveness property, which can be cast as a safety property:

$$(p_0 \in [-0.59, -0.4]) \Rightarrow (k = 200 \Rightarrow (p_k \geq p^* \wedge r_k \geq 90)),$$

where $p^* = 0.45$ is the goal position. Note that the 200-step bound on k was found to be conservative and could be changed depending on the controller. Moreover, for the composite noise system, we verify over the range $p_0 \in [-0.55, -0.4]$ given the additional scalability cost of having both contrast and blur networks present in the verification loop and given the controller is less robust for starting positions nearer -0.59.

Since computing reachable sets for entire initial range results in too much uncertainty, we split in the initial range into sub-intervals of size 0.00025 and verify each sub-interval as a separate verification instance. If an instance did not terminate within 24 hours, we split that interval into five sub-intervals and repeated. All instances are verified safe; summary statistics are provided in Table 1. Note the increased verification time for the Blur vs the Contrast models is due to using a cluster-based computing resource for Blur instead of the servers mentioned in the body of the paper.

Appendix B. F1/10 Car

This section explains the F1/10 case study.

B.1. Dynamics

The car dynamics are borrowed from Ivanov et al. (2020a) and are assumed to follow a bicycle model (with no slipping/friction) as follows:

$$\begin{aligned} \dot{x_1} &= vcos(\theta) \\ \dot{x_2} &= vsin(\theta) \\ \dot{v} &= -c_a v + c_a c_m (u - c_h) \\ \dot{\theta} &= \frac{V}{l_f + l_r} tan(\delta), \end{aligned} \tag{5}$$

where v is linear velocity, θ is orientation, x_1 and x_2 are the two-dimensional position; $u = 16$ is the throttle input, and δ is the heading input (ranging between -15 and 15 degrees); c_a is an acceleration constant, c_m is a car motor constant, c_h is a hysteresis constant, and l_f and l_r are the distances from the car's center of mass to the front and rear, respectively. Parameter values were identified by Ivanov et al. (2020a) as: $c_a = 1.633, c_m = 0.2, c_h = 4, l_f = 0.225m, l_r = 0.225m$. Note that the car model is in continuous time; however, the controller is sampled at discrete time steps (at 10Hz), the system effectively operates in discrete time.

B.2. Canonical LiDAR Model

The canonical LiDAR model was developed by Ivanov et al. (2020a) for a square track with width 1.5m and length 20m. The LiDAR model is hybrid since there are four walls that each LiDAR ray could possibly hit:

$$y_k^i = g_c(x_k, y_k, \theta_k) = \begin{cases} d_k^r / cos(90 + \theta_k + \alpha_i) & \text{if } \theta_k + \alpha_i \leq \theta_r \\ d_k^b / cos(180 + \theta_k + \alpha_i) & \text{if } \theta_r < \theta_k + \alpha_i \leq -90 \\ d_k^t / cos(\theta_k + \alpha_i) & \text{if } -90 < \theta_k + \alpha_i \leq \theta_l \\ d_k^l / cos(90 - \theta_k - \alpha_i) & \text{if } \theta_l < \theta_k + \alpha_i, \end{cases} \tag{6}$$

where $d_k^t, d_k^b, d_k^l, d_k^r$ are distances to the four walls; the α_i denote the relative angles for each ray with respect to the car's heading, i.e., $\alpha_1 = -115, \alpha_2 = -103.5, \ldots, \alpha_{21} = 115$; θ_l and θ_r are the angles to the left and right corners of the current hallway, respectively. Please consult Ivanov et al. (2020a) for a full model description.

B.3. Controllers

We use the 12 pre-trained controllers from Ivanov et al. (2020a). These controllers were trained using deep deterministic policy gradients (DDPG, Lillicrap et al. (2015)) and twin delayed DDPG (TD3, Fujimoto et al. (2018)). The controllers are end-to-end, i.e., each controller $h_c : y_k \mapsto \delta_k$ takes a LiDAR scan and outputs a heading. All controllers were verified by Ivanov et al. (2020a) to be safe for the canonical LiDAR model, yet all controllers resulted in at least some crashes when used on the real platform.

B.4. Training Set

The training set consists of 115 real-data trajectories (roughly 10 trajectories per each of the 12 controllers). The original trajectories contain only (full 1081-dimensional) LiDAR scans. Since

the proposed data-driven modeling approach requires car states as well, we used a particle filter (as presented by Thrun et al. (2005)) to estimate the car poses. Given the state estimates, the final labeled dataset is $\mathcal{D} = \{(x_{1,i}, x_{2,i}, \theta_i, y_i)\}$, where $(x_{1,i}, x_{2,i})$ denote the car's (x, y)-coordinates and θ_i is the car's orientation.

B.5. Adversarial LiDAR Model

To train the LiDAR noise model, we label each LiDAR scan according to the procedure in Section 4. Given a canonical scan $g_c(x_{1,i}, x_{2,i}, \theta_i)$, we set $n_i^d = 1$ if $|g_c(x_{1,i}, x_{2,i}, \theta_i)^d - y_i^d| > t_\Delta^*$, and $n_i^d = 0$, otherwise. The threshold t_Δ^* is chosen to minimize the error between noised canonical scans and real scans over the training data. Using the labeled training set $D_l = \{((g_c(x_{1,i}, x_{2,i}, \theta_i), x_i), n_i)\}$, we train classifiers of increasing sizes (listed in Table 3) using weighted BCE (to account for class imbalance between adversarial and non-adversarial examples). The adversarial example (class 1) loss is weighted 4 times the class 0 loss.

Figures 11 show an evaluation of the adversarial model used the control-based metric, over most controllers. The figures illustrate that the classifier-based adversarial model captures most of the control-action variability due to the adversarial noise (especially when compared with the canonical model).

As further evaluation, Figures 12 also show the evaluation using the trajectory-based metric. Notice that overall the trajectories look quite similar; in particular most crashes observed in the real world correspond to crashes observed in simulation, too.

Noiser Training Results					
Model Size	3x100	4x100	5x100	6x100	6x200
WSA [%]	61	77	87	98	100
Control Err.	3700	2312	1485	774	594
Safe Trajectory Distribution [%]					
Controller: **Real Data**	**Simulated Data Using Noiser Model**				
DDPG, 64x64, 1: **0**	19	25	16	15	8
DDPG, 64x64, 2: **20**	31	43	31	25	20
DDPG, 64x64, 3: **80**	88	83	88	82	97
DDPG, 128x128, 1: **80**	75	67	69	62	63
DDPG, 128x128, 2: **40**	89	84	88	85	88
DDPG, 128x128, 3: **0**	29	29	29	32	27
TD3, 64x64, 1: **90**	78	79	64	82	100
TD3, 64x64, 2: **90**	75	80	92	77	74
TD3, 64x64, 3: **90**	94	94	97	72	100
TD3, 128x128, 1: **90**	100	100	95	96	100
TD3, 128x128, 2: **90**	85	78	98	91	91
TD3, 128x128, 3: **90**	87	91	95	86	100

Table 3: Training results and crash distributions for different noise models. Pre-trained controllers and real data were borrowed from Ivanov et al. (2020a).

B.6. Denoiser

The denoiser's goal is to take a noised LiDAR scan and convert it to a canonical scan. We train a 1×100 fully-connected denoiser NN, $h_d : y_i \mapsto \hat{y}_i$, on pairs $(g_c(x_{1,i}, x_{2,i}, \theta_i), y_i)$ using BCE loss and weight decay.

B.7. Full System

The full system is described below:

$$\begin{aligned} x_{1,k+1}, x_{2,k+1}, v_{k+1}, \theta_{k+1} &= f(x_k, y_k, v_k, \theta_k, \delta_k) \\ y_k &= g_c(x_{1,k}, x_{2,k}, \theta_k) \\ y_k^a &= g_n(x_{1,k}, x_{2,k}, \theta_k, y_k) \\ \hat{y}_k &= h_d(y_k^a) \\ \delta_k &= h_c(\hat{y}_k), \end{aligned} \tag{7}$$

where f is obtained by following the continuous-times dynamics in (5) for 0.1s.

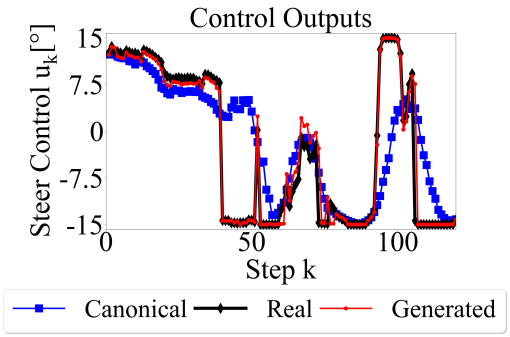
B.8. Verification

We verify that the car can navigate the first right-hand turn. In particular, the car is started in a .2 m interval in the middle of the first segment (centered at the origin), 9m from the front wall ($x_{2,0} = 1$). We verify that the car reaches 2.5m from the left wall in the second hallway (since that is roughly how the where the real data ends, i.e., the generative model is not reliable beyond that point). Similar to the MC case, we formulate the liveness property as a bounded safety property:

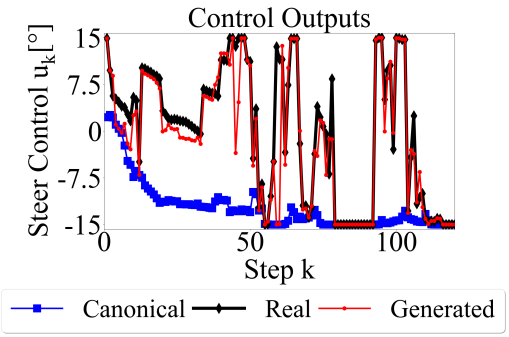
$$(x_{1,0} \in [-0.1, 0.1]) \Rightarrow (\phi(x_{1,k}, x_{2,k}) \wedge (k = 70 \Rightarrow x_{1,k} \geq 1.75)),$$

where $\phi(x_{1,k}, x_{2,k})$ is the safety property of no crashes.

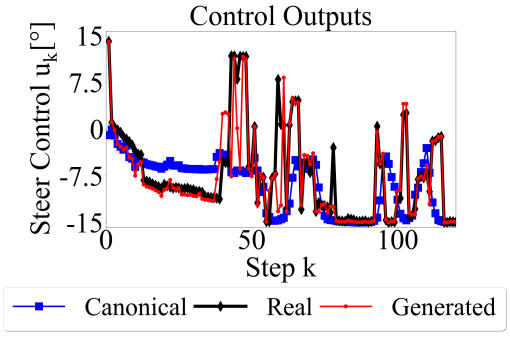
Verifying the F1/10 car safety is particularly challenging both due the complex LiDAR model and due to the adversarial noise classifier. All of these components introduce possible branching in model: if the uncertainty is too high, a LiDAR ray could reach different walls, in which case both models need to be verified separately. Similar to the case of MC, we split the initial range into sub-intervals of size 0.000025 and sub-divide each interval if verification takes longer than 24 hours. The minimum interval sized used was 0.000001. Summary statistics are provided in Table 1.



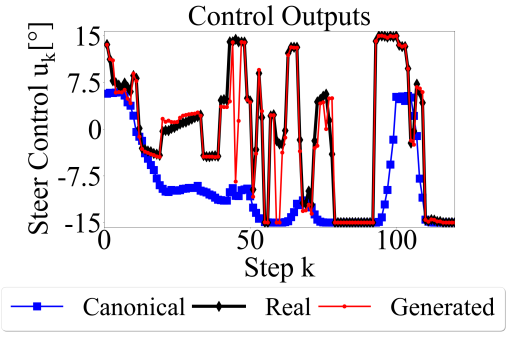
(*a*) DDPG, 64, 1



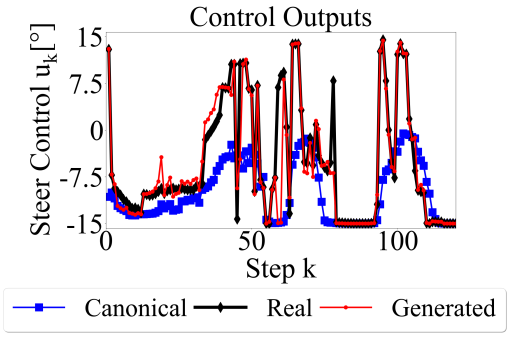
(*b*) DDPG, 64, 2



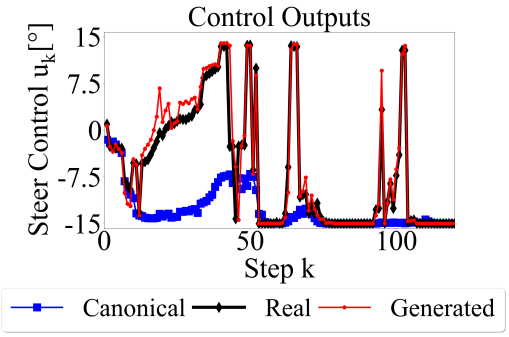
(*c*) DDPG, 64, 3



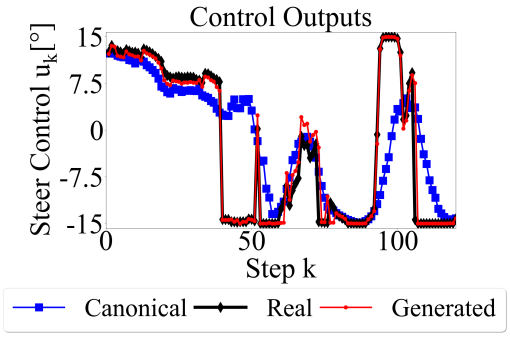
(*d*) DDPG, 128, 1



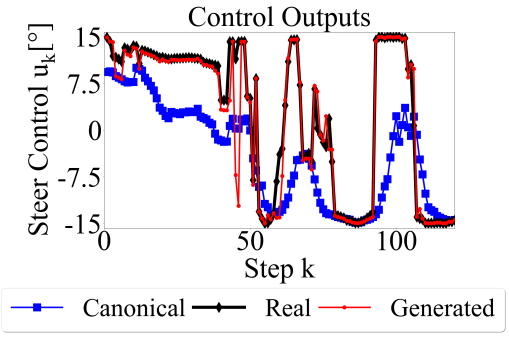
(*e*) DDPG, 128, 2



(*f*) DDPG, 128, 3

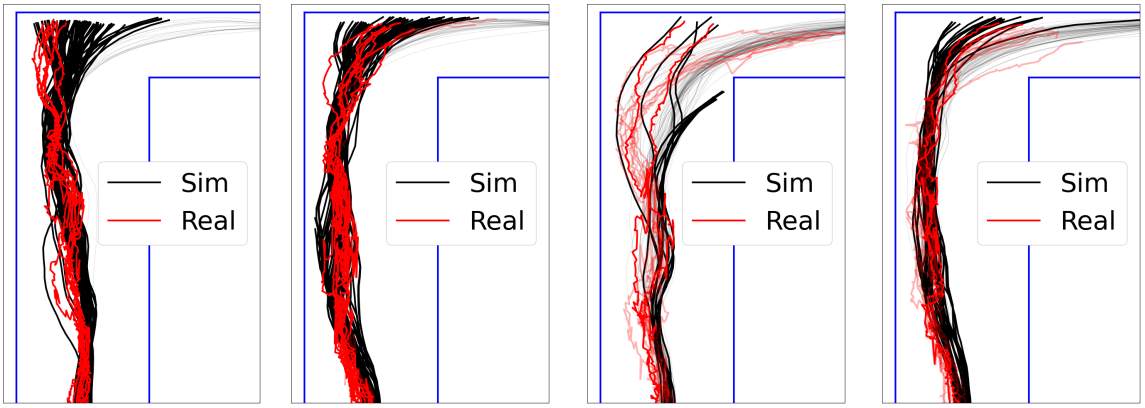


(*g*) TD3, 64, 1

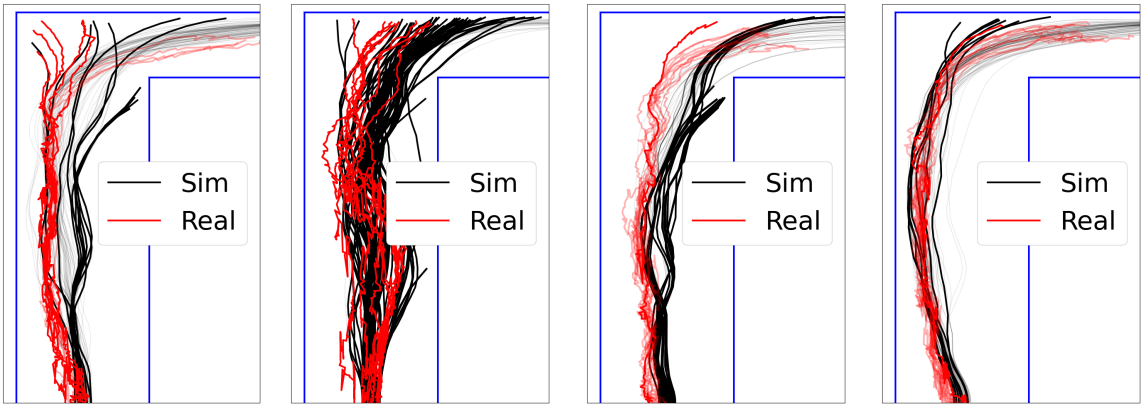


(*h*) TD3, 64, 2

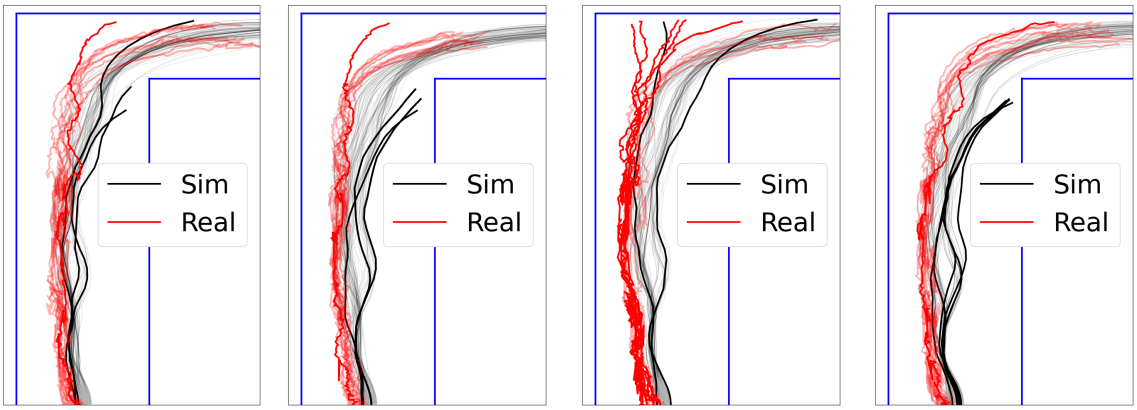
Figure 11: Control-based evaluation of F1/10 noise model, using 10 controllers from Ivanov et al. (2020a). Figures outputs from each controller on a training trajectory for canonical, noised and real measurements.



(*a*) DDPG, 64, 1 (*b*) DDPG, 64, 2 (*c*) DDPG, 64, 3 (*d*) DDPG, 128, 1



(*e*) DDPG, 128, 2 (*f*) DDPG, 128, 3 (*g*) TD3, 64, 1 (*h*) TD3, 64, 2



(*i*) TD3, 64, 3 (*j*) TD3, 128, 1 (*k*) TD3, 128, 3 (*l*) TD3, 128, 3

Figure 12: Simulated trajectories under the 5x100 noiser model vs. real trajectories for all twelve controllers from Ivanov et al. (2020a) (without using a denoiser). Bold trajectories indicate those that ended in a crash.